

EXPERT SYSTEMS FOR C3I VOLUME 1 A USER'S INTRODUCTION
(U) NITRE CORP BEDFORD MA J A CLAPP ET AL OCT 85
ESD-TR-85-125 F19628-84-C-0001

(U) MITRE CORP BEDFORD MA J A CLAPP ET AL OCT 85
ESD-TR-85-125 F19628-84-C-0001

F/G 17/2

NL

[illegible]



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12

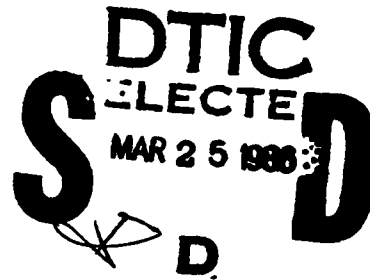
Expert Systems for C³I

Volume I

A User's Introduction

AD-A165 693

MITRE SOFTWARE CENTER



DTIC FILE COPY

J. A. Clapp
S. M. Hockett
M. J. Prella
A. M. Tallant
D. D. Triant

October 1985



Prepared for Deputy for Acquisition Logistics and
Technical Operations, Electronic Systems Division,
AFSC, United States Air Force, Hanscom Air Force
Base, Massachusetts.

Approved for public release; distribution unlimited.

86 3 21 015
MITRE

Bedford, Massachusetts

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.


REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.



GEORGE G. JACKELEN, Major, USAF
Project Officer, Project 5720
Computer Technology and Support Division

FOR THE COMMANDER



ROBERT J. KENT
Director, Computer Systems Engineering
Deputy for Acquisition Logistics
and Technical Operations

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				
1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MTR-9630 ESD-TR-85-125		5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION The MITRE Corporation		6b. OFFICE SYMBOL (If applicable)		7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Burlington Road Bedford, MA 01730		7b. ADDRESS (City, State and ZIP Code)		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Deputy for Acquisition (cont)		8b. OFFICE SYMBOL (If applicable) ALSE		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-84-C-0001
8c. ADDRESS (City, State and ZIP Code) Electronic Systems Division, AFSC Hanscom AFB, MA 01731-5000		10. SOURCE OF FUNDING NOS.		
11. TITLE (Include Security Classification) EXPERT SYSTEMS FOR C ³ I. (continued)		PROGRAM ELEMENT NO.	PROJECT NO. 5720	TASK NO.
12. PERSONAL AUTHOR(S) Clapp, J.A.; Hockett, S.M.; Prella, M.J.; Tallant, A.M.; Triant, D.D.		WORK UNIT NO.		
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr. Mo., Day) 1985 October		15. PAGE COUNT 61
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB. GR.		
		Artificial Intelligence (AI)		
		Expert Systems		
		Expert Systems Tutorial		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This report was written in response to the growing interest at ESD and MITRE in developing expert systems for C ³ I. This document is intended to help potential users of expert systems technology to understand what expert systems are and what they can do. It includes a brief tutorial on expert systems, a review of the expert systems component of DARPA's Strategic Computing Program, and a survey of expert system building tools.				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input checked="" type="checkbox"/> DTIC USERS <input type="checkbox"/>		21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL Diana F. Arimento		22b. TELEPHONE NUMBER (Include Area Code) (617)271-7454		22c. OFFICE SYMBOL Mail Stop D230

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

8a. Logistics and Technical Operations.

11. VOLUME I: A USER'S INTRODUCTION

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

ACKNOWLEDGMENTS

This document has been prepared by The MITRE Corporation, Software Center General Support, under Project No. 5720, Contract No. F19628-84-C-0001. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts 01731.

Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	



TABLE OF CONTENTS

<u>Section</u>		<u>Page</u>
1	OVERVIEW	1
	1.1 PURPOSE OF THE REPORT	1
	1.2 APPROACH	1
2	EXPERT SYSTEMS AND THEIR USES	3
	2.1 WHAT IS AN EXPERT SYSTEM?	3
	2.2 THE ANATOMY OF AN EXPERT SYSTEM	5
	2.2.1 The Knowledge Base	5
	2.2.2 The Inference Engine	8
	2.2.3 The User Interface	10
	2.3 CHOOSING AN EXPERT SYSTEM APPLICATION	12
	2.3.1 Advantages of Expert Systems	12
	2.3.2 A Word of Caution	13
	2.3.3 Selecting the Right Application	14
3	EXAMPLES OF FIELDDED EXPERT SYSTEMS	17
	3.1 ACE	17
	3.2 DENDRAL	19
	3.3 MACSYMA	19
	3.4 MYCIN	19
	3.5 ONCOCIN	20

<u>Section</u>	<u>Page</u>
3.6 PROSPECTOR	20
3.7 PUFF	21
3.8 SOPHIE	21
3.9 XCON	22
4 TOOLS FOR BUILDING EXPERT SYSTEMS	23
4.1 THE ROLE OF TOOLS IN BUILDING EXPERT SYSTEMS	24
4.2 FEATURES OF EXPERT SYSTEM BUILDING TOOLS	24
4.3 PROGRAMMING LANGUAGES FOR WRITING EXPERT SYSTEMS	27
4.3.1 LISP	28
4.3.2 PROLOG	28
4.3.3 SMALLTALK	29
4.4 DESCRIPTION OF TOOLS	29
4.4.1 EMYCIN	29
4.4.2 MRS	32
4.4.3 OPS5	33
4.4.4 YAPS	33
4.4.5 ROSIE	33
4.4.6 LOOPS	34
4.4.7 AGE	34
4.4.8 KES	35
4.4.9 TIMM	35

<u>Section</u>	<u>Page</u>
4.4.10 DUCK	36
4.4.11 KEE	36
4.4.12 S.1	37
4.4.13 M.1	37
4.4.14 EXPERT-EASE	38
5 DARPA'S STRATEGIC COMPUTING PLAN	39
5.1 GOALS	39
5.2 OVERVIEW OF THE SCP	40
5.3 EXPERT SYSTEM TECHNOLOGY	41
5.4 APPLICATIONS	44
5.4.1 Autonomous Vehicles	44
5.4.2 Pilot's Associate	45
5.4.3 Battle Management System	45
5.5 HARDWARE	46
5.6 MANAGEMENT, SCHEDULE AND COST	46
6 CONCLUSIONS AND RECOMMENDATIONS	49
6.1 CONCLUSIONS	49
6.2 RECOMMENDATIONS	50
LIST OF REFERENCES	51

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Page</u>
2-1 Knowledge-Based System vs. Conventional Software System	6
2-2 Inheritance--An Example	7

LIST OF TABLES

<u>Table</u>	<u>Page</u>
2-1 Sample User-Machine Dialogue for the STAMMER2 Production System	11
3-1 Examples of Fielded Expert Systems	18
4-1 Expert System Building Tools: General Information	30
4-2 Expert System Building Tools Characteristics	31
5-1 Strategic Computing Expert Systems Technology Major Functional Capabilities and Milestones	43
5-2 Strategic Computing Plan Cost Summary in \$Millions	47
5-3 Technology Panel on Strategic Computing Joint Directors of Laboratories Subpanel Membership	48

SECTION 1

OVERVIEW

1.1 PURPOSE OF THE REPORT

There has been a tremendous burgeoning of interest in artificial intelligence (AI) over the last few years. Investments of commercial and government sponsors reflect a widespread belief that AI is now ready for practical applications. The area of AI currently receiving the greatest attention and investment is expert system technology. Most major "high tech" corporations have begun to develop expert systems, and many software houses specializing in expert system tools and applications have recently appeared.

The defense community is one of the heaviest investors in expert system technology, and within this community one of the application areas receiving greatest attention is C³I. Many ESD programs are now beginning to ask whether expert system applications for C³I are ready for incorporation into ESD-developed systems, and, if so, what are the potential benefits and risks of doing so.

This report was prepared by the MITRE Software Center to help ESD and MITRE personnel working on acquisition programs to address these issues and to gain a better understanding of what expert systems are all about. Many surveys and reports on expert systems have been written, but most focus on the technology itself and the various construction techniques it uses. In this report we have tried instead to provide the kind of information needed by potential users of expert system technology, to help them determine whether this new technology is appropriate for their applications. Thus, it is written from a user's, rather than a developer's perspective.

1.2 APPROACH

The primary intention of this report is to investigate what expert systems are and the advances that are being made in expert system technology for C³I applications. The report begins with a brief tutorial on expert systems, emphasizing how they differ from conventional software systems and what they are best at doing. (section 2). Some examples of expert systems are presented in section 3, to let the reader see by example how some expert systems perform and what kinds of applications are being addressed. Section 4 discusses the use of tools for building expert systems and

describes some of the tools now available. Section 5 looks toward the more distant future by describing the Defense Advanced Research Projects Agency's (DARPA) Strategic Computing Initiative and its potential impact on the development of "next generation" expert systems. Finally, section 6 presents the conclusions of the report.

SECTION 2

EXPERT SYSTEMS AND THEIR USES

Considering all of the attention they have received, it is surprisingly difficult to find a widely accepted, concrete definition of what expert systems are, how they differ from so-called conventional software systems, and what kinds of applications the current state of expert system technology is best suited to perform. The goal of this section is to shed some light on these questions. The section also describes several operational expert systems and takes a brief look at their performance.

2.1 WHAT IS AN EXPERT SYSTEM?

The most commonly offered definition of an expert system is "a system that performs like an expert." This definition is somewhat misleading--no expert systems can completely replace human beings at their jobs, nor were they designed to do so. Current expert systems should be more accurately described as capable assistants which help their human counterparts to perform tasks that require in-depth understanding of the problem domain and flexible, sophisticated problem-solving capabilities.

Expert systems solve problems by using knowledge of the problem, characteristics of acceptable solutions, and methods or rules for problem solving. Conventional systems sometimes require the same knowledge, but it is embedded in the algorithms used by the software. The system behavior is explicitly programmed for every combination of inputs that is expected to occur. Any changes require reprogramming the system. An expert system is usually constructed to contain a data base representing the knowledge it needs for problem solving. It uses general reasoning processes, such as induction, to create new facts from existing facts in its data base. It may also contain information or "rules of thumb" about reasoning for problem solving in a specific application. This gives expert systems the potential for dealing with situations that were not foreseen when the system was programmed. The knowledge in an expert system might be changed by the system itself or (more commonly) by the user without reprogramming the software. This allows an expert system to "learn" and allows users to "teach" a system by modifying the rules for its behavior.

Typically, a task that can be performed according to an algorithm does not require human expertise--anyone who follows the algorithm can perform the job. Likewise, such tasks do not need

expert systems. In contrast, an expert brings to his job a wealth of knowledge obtained through experience. This experiential knowledge usually includes:

- o detailed, fragmentary information about special cases,
- o partial, generalized knowledge or "rules of thumb,"
- o qualitative rather than quantitative reasoning,
- o conceptual models expressing the relationships among problem components, and
- o knowledge about the reliability of the information being used.

For example, compare how a novice and an expert automobile driver might choose their driving speeds.

Novice: Drive at the speed limit.

If it is foggy or if there is precipitation on the road, subtract 10 mph from speed.

Expert: Drive at approximately the speed limit.

If the road is wet, slow down a little.

If the road is icy, slow down a lot, but if the road is sanded, you can go a little faster than otherwise.

If you see children near the road, drive more slowly.

Usually drive more slowly on an unfamiliar road than on a familiar road, unless the unfamiliar road is a highway.

If you are in a hurry, you can probably drive 5 to 10 mph over the speed limit without getting stopped, except watch out for a speed trap near the bridge on Oak Street.

Go very slowly when approaching the curve on Main Street between Willow and Pine.

This latter kind of knowledge is not well-suited to an algorithmic computer program, but expert systems employ a variety of new programming techniques that can make direct use of this kind of information. To understand how expert systems can do this, it is

useful to consider how they are built and how they differ from conventional software systems.

2.2 THE ANATOMY OF AN EXPERT SYSTEM¹

A conventional computer program typically consists of a coded algorithm for calculating answers to queries it receives, and a data base of objects and their features (attributes) that is accessed by the algorithm to provide data for its calculations. The algorithm embodies most of the knowledge about the problem area that the system uses. In place of a coded algorithm, an expert system has two components: a knowledge base that stores explicit knowledge about the problem in the form of rules and relations, and a general purpose reasoning mechanism known as the inference engine that uses the knowledge in the knowledge base to solve problems (see figure 2-1).

2.2.1 The Knowledge Base

The knowledge base contains information about objects in the system's domain and rules affecting those objects. Information about objects may be of two types: facts about generic objects, their characteristics and relations to other objects (e.g., a car has wheels and is a type of moving object), and facts about the current state of the world (e.g., my car is parked in the driveway). Information of both types is often stored in special structures (sometimes known as frames), which are used for recording objects, their associated attributes, and sometimes procedures attached to that object (e.g., a procedure for changing a tire might be attached to the "car" object) or constraints affecting that object. Many systems have a mechanism by which frames can inherit attributes from other frames to which they are related (e.g., a Chevrolet is a car and, therefore, a Chevrolet has wheels) (see figure 2-2). This is a powerful way to store data, since it minimizes the number of attributes that must be directly coupled to an object and the number of changes to the data base needed when an attribute changes.

¹To be more precise, the "anatomy" described in this section characterizes a larger class of systems known as knowledge-based systems. Expert systems are knowledge-based systems that perform some of the functions of a human expert by capturing and using experts' knowledge. However, since expert systems are by far the most common and best known types of knowledge-based systems, people tend to ignore the distinction. We shall do the same!

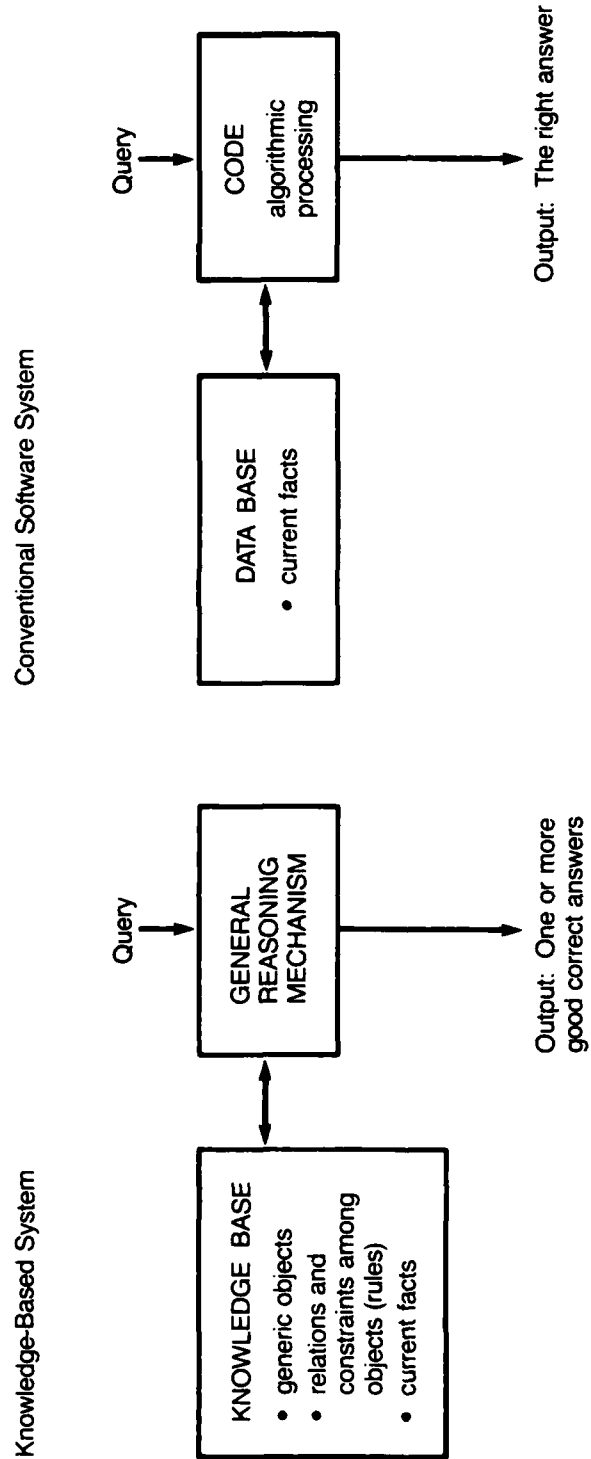


Figure 2-1. Knowledge-Based System vs. Conventional Software System

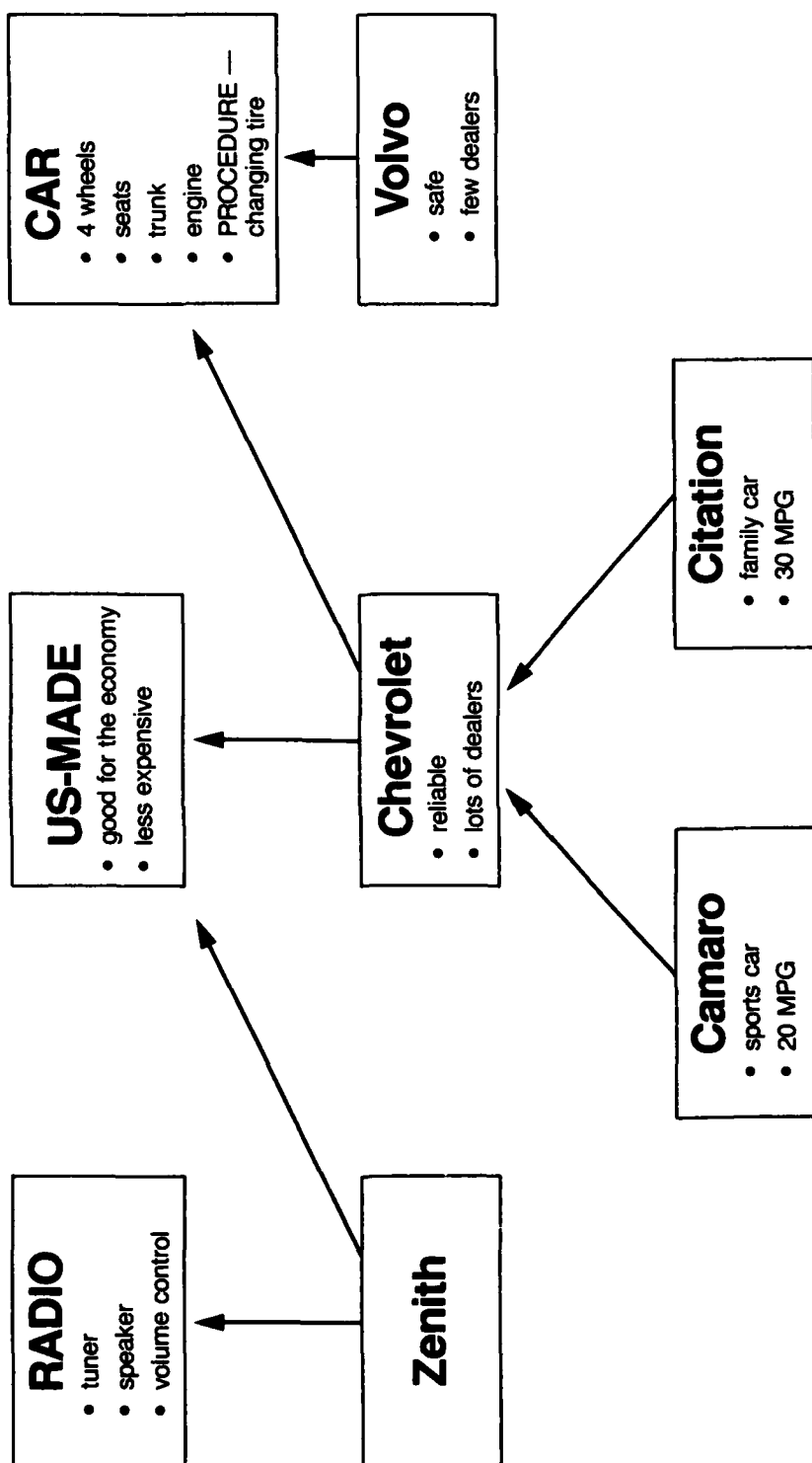


Figure 2-2. Inheritance — An Example

Knowledge bases usually also contain rules describing how to change from one problem state to another, either by changing a value in the data base or by generating actions to be taken. Rules can cause an action or procedure to be initiated (e.g., "if the time is after 6 p.m., then change the time of day to 'night'"). Rules can also be used to derive new information (e.g., "if a person is a parent and a female, then that person is a mother"). Other rules may express constraints that do not allow certain states or combinations of values to occur. For example, there may be a rule which says that only one object can be at the same location at the same time. This rule might be used to reject a potential solution, generated by an expert system, which moved one piece to the location of another in a chess game or moved a piece of cargo to an already occupied space in a cargo loading problem. Similar constraints might also be used to check the validity of user inputs as well as values generated by the system. For example, the expert system might reject an illegal move in a chess game, whether by the player or the system. Constraints can also be used to monitor conditions for which an action is required (e.g., if the difference between one observation and another is greater than some specified value, then an operator should be alerted).

Another kind of rule might contain a hypothesis and conclusion, e.g., "if there's smoke, there is an 80% chance there is a fire." These rules can be used to suggest solutions, even when they are uncertain. Such rules can embody the "hunches" of experts.

2.2.2 The Inference Engine

The inference engine does the work of transforming input states or problems into output states, corresponding to goals or solutions. Individual rules in the knowledge base define single steps, while solving the problem may involve many steps or transformations. For rule-based expert systems (the most common type), the process can be characterized as searching for a match between rules in the knowledge base and either the current state or the current goals of the problem, selecting a rule which matches, and executing that rule to create a new state or a subproblem until all necessary transformations have been found. It has been likened to finding a path through a maze that has many dead ends. In the expert system, the paths lead among states in the system for which allowable transformations have been specified in the knowledge base.

One of the significant differences among expert systems is the method by which transformations are selected. The inference engines of rule-based systems have a general logic for selecting which rules to try, for testing whether a rule is ready to fire, and for

choosing which rule to fire from among those that are ready. Some systems search forward from the initial state toward a goal state (known as forward chaining), some search back from a hypothesized goal state toward the initial state (known as backward chaining), and some do both. The optimum method may be a function of the problem, which may have fewer options to match in one direction than another so the system can converge more quickly on the matches that have to be tried. It is naturally desirable to try as few rules as possible, and only those that have a chance at leading to a solution.

The definition of when a rule is ready to fire depends on whether forward or backward chaining is being used. In a forward chaining system, rules are fired when the antecedents, or if-part, are satisfied. Such a system may be thought of as data-driven. For example, consider the following rule: "if precipitation and temperature >32 degrees F, then rain." This rule will be ready to fire in a forward chaining system if the system "believes" that the value of temperature is greater than 32 and that precipitation is present. If this rule actually fired, then the system would "believe" that it is raining. In a backward chaining system, rules are fired when the consequent, or then-part, needs to be verified. Such a system may be thought of as goal-driven. The example rule will be ready to fire in a backward chaining system if the system needs to verify that it is raining. If this rule actually fired, then the system would next try to verify that the temperature was greater than 32 and that precipitation was present. In both cases, firing one rule will normally establish one or more other rules as ready to be fired.

If more than one rule is ready to fire at the same time, a method is needed to choose which rule to fire. This process is known as conflict resolution. Different systems use different approaches to conflict resolution methods. For example, one approach is to simply select the first rule encountered, while another is to select the rule with the greatest number of "if" conditions (i.e., the most specific rule).

If no more rules can be fired, then a final goal state has been reached, or the system has reached a dead-end, or the problem cannot be solved with the knowledge in the system. If a dead-end has been reached, the current problem state or the current goal must be changed in order for the system to be able to explore a different path toward a solution. The process of backing off from a dead-end is known as "backtracking." A common way of backtracking is to undo the most recent transformations applied to the problem in order to revert to an earlier state. General techniques and domain-specific techniques can make this process of "backtracking" more efficient.

2.2.3 The User Interface

Expert systems are expected to behave intelligently. This includes their form of communication with the user. Interfaces are needed for specifying and constructing the knowledge base, and for conducting a dialogue with the end user during the problem solving process. If the interface is between the system and a human, then the interface may include natural language for input and/or output. Graphic inputs or outputs may also be appropriate. Any form of communication that fosters understanding should be employed. Natural language presents the largest technical challenge. No full-fledged natural language interfaces (i.e., ones that can understand language as well as humans can) are yet available, but many expert systems successfully employ pseudo-natural language interfaces (i.e., ones in which the form and content of sentences are restricted).

The knowledge-based structure of an expert system allows its user-machine interface to contain an especially powerful feature--the ability to explain the system's results. An explanation facility can greatly increase user confidence in the system's answers and can serve as a powerful debugging tool for system developers.

An example of user-machine dialogue for a rule-based C³I expert system is shown in table 2-1. The sample dialogue comes from the STAMMER2 Production System for Tactical Situation Assessment. STAMMER2 concentrates on the specific task of merchant detection from radar and external messages.

Some comments about the dialogue are in order. First, the system speaks with the user in English (see, for example, lines 3 and 4). Second, the system answers questions about why a fact was used or how it reached its conclusion. For example, in line 4, STAMMER asserts (assertion A0223) that CONTACT2 is somewhat unlikely to be a MERCHANT. In line 5, the user types "WHY is A0223" in order to learn the immediate reasons for STAMMER concluding A0223. In lines 6 and 7, STAMMER responds with the rule that was used, namely, CLOSE-POPUP. In line 8, the user asks how the rule was applied. STAMMER responds with a list of assertions that were involved in permitting the rule to help conclude the assertions A0233.

The following is dialogue between a user and the STAMMER2 Production System for Tactical Situation Assessment. The system concentrates on the task of merchant detection.

Table 2-1. Sample User-Machine Dialogue for the STAMMER2 Production System

1. **RADAR contact at (63.67 - 24.17) Time: 115**
2. **Associated with track CONTACT2**
3. Report: CONTACT2 was sighted in the merchant lane LANE2
4. A0223: CONTACT2 is somewhat unlikely to be (-.19) a MERCHANT
5. Question? **WHY is A0223**
6. STAMMER applied the rule(s)
7. CLOSE-POPUP
8. Question? **HOW does rule CLOSE-POPUP apply to A0223**
9. The rule was applied with the assertions
10. A0215: CONTACT2 is a contact
11. A0214: SIGHTING3 IS DEFINITELY (.99) THE FIRST SIGHTING OF CONTACT2
12. A0220: 11.73514 is the range of SIGHTING3
13. A0222: 11.73514 is less than 12
14. Question? **Quit**
15. Leaving EXPLAIN
16. **RADAR contact at (63.74 - 24.25) Time: 125**
17. **Associated with track CONTACT2**
18. Report: CONTACT2 was sighted in the merchant lane LANE2
19. A0223: CONTACT2 is somewhat unlikely to be (-.39) a MERCHANT
20. Question? **WHY is A0223**
21. STAMMER applied the rule(s)
22. FASTER-THAN-A-MERCHANT CLOSE-POPUP
23. Question? **HOW does rule FASTER-THAN-A-MERCHANT apply to A0223**
24. The rule was applied with the assertions
25. A0215: CONTACT2 is a contact
26. A0239: SIGHTING4 is a sighting of CONTACT2
27. A0244: SIGHTING4 is other than a first sighting of its platform
28. A0247: 28.26315 is the speed of SIGHTING4
29. A0257: 28.26315 is greater than 25
30. Question? **Quit**
31. Leaving EXPLAIN

Source: McCall, et al., p. 27 f. Quoted with permission.

The run begins with the user entering data about a sighting. "Time" is the present simulated time. STAMMER asserts that the contact is somewhat unlikely to be a merchant. It provides an explanation at the request of the user.

Material entered by the user is in bold face type.

Finally, observe that STAMMER states the confidence associated with its assertions, as in line 4, and that the confidence of an assertion is dynamic. That is, confidence varies with time. In the sample run, the confidence associated with the assertion that CONTACT2 is not a MERCHANT increases from -.19 (line 4) to -.39 (line 19) as new information is made available in lines 16 and 17.

Confidence can assume negative or positive values. Assertions satisfying a NOT condition are associated with negative confidence when the inverse assertion is associated with positive confidence. Thus, the assertion NOT A0223 (which translates into "not a merchant") is associated with a confidence of -.19 (line 4) and -.39 (line 19).

2.3 CHOOSING AN EXPERT SYSTEM APPLICATION

Expert system technology has opened the door to automating tasks that previously could not be considered for automation. On the other hand, expert systems cannot solve all problems. Identifying an appropriate problem for an expert system solution is a crucial first step in developing a successful application. This is the subject of this section.

2.3.1 Advantages of Expert Systems

The knowledge-based architecture of expert systems provides a number of advantages over conventional systems. For applications where one or more of these benefits are particularly important, the application may be worth considering for an expert system solution.

These advantages include the following:

1. Expert systems can deal with fuzzy, uncertain, or incomplete data. This is the kind of information people typically have to use in carrying out their day-to-day activities, but conventional systems usually balk at. Though accurate data is always desirable, in the real world precise information is often unavailable. For example, in military planning the enemy's situation and intentions are rarely known with accuracy. Expert systems can use various problem-solving approaches to handle information of this type, such as heuristic reasoning and techniques for using and combining uncertain evidence.

2. Expert systems can incorporate fragmentary or "special case" information. Information can be added to the knowledge base without knowing when or how that information will be invoked to solve a problem. In a conventional program, rules and relations must be explicitly fitted into the problem-solving algorithm.
3. Expert systems are easier to modify than conventional systems. Knowledge expressed as separate rules is far more modular than knowledge embedded in algorithms, and, consequently, is easier to change. Rules can generally be added, deleted, or modified relatively independently, without affecting the rest of the program.
4. Expert systems can explain their answers. By unraveling the reasoning by which it reached an answer, a knowledge-based system can describe its thought process as a series of English-like rules. This benefits the end user by increasing his understanding of the result, instilling confidence, and if more than one solution is possible, helping him to select among solutions by presenting the pros and cons of each alternative.

2.3.2 A Word of Caution

Expert systems have many advantages, but they also present some risks. These must also be considered in selecting an application. Expert system technology is still new, and it carries the same risks as does any new technology. There is a limited set of people available who know how to design, build, and maintain expert systems. The system acquisition process for expert systems is still unexplored territory. Little is known about how to integrate expert systems into larger systems.

The special characteristics of expert systems bring with them some additional risks. Configuration control may be more difficult for expert systems than for conventional software systems because of the changeable nature of the knowledge base, a feature that at the same time represents one of the greatest strengths of expert systems. Configuration control may also be hampered by the unspecified relations among knowledge base elements and by the deliberate use of side effects in many system designs. Finally,

rigorous testing against specifications may prove difficult for some types of applications, such as those for which the domain knowledge is fuzzy, uncertain, or incomplete.

2.3.3 Selecting the Right Application

There are no simple rules for picking the right applications for expert systems--every application has its own unique problems and opportunities that affect both the desirability of an expert system solution and the chances of its success. Even applications that have been carefully selected for building an expert system usually begin with a prototyping phase to verify the feasibility of the application.

Nonetheless, some general indicators can be considered in the first stage of selecting an application. The following are some features that suggest a problem may benefit from an expert system solution:

1. Characteristics of Data:

The data on which the application is based are incomplete, imprecise, fragmentary, or too complex to organize into a unified model.

2. Characteristics of the Problem-Solving Method:

There is no practical algorithmic procedure for solving the problem.

Solutions require reasoning with uncertain evidence.

3. Characteristics of the Problem:

There are many possible solutions but few acceptable ones, or there are no completely correct solutions.

The system will be modified frequently.

Solutions change with time and context.

The system needs to be highly flexible in the kinds of questions it can answer.

For problems that are likely to benefit from expert system technology, the following constraints should also be met to improve the chances that an expert system approach will be successful:

1. The problem and problem-domain are well bounded. It is important to "think small" in choosing an initial application, and expand incrementally after an initial success has been achieved.
2. The problem is neither too easy nor too difficult for human experts. A useful gauge of this is that the problem should take on the order of an hour or two to solve, usually not minutes, and definitely not days.
3. There is at least one expert in the problem domain available to work closely with the system developers. Developing expert systems is a cooperative effort between the domain experts and the experts in knowledge-based systems--seldom can either group build a successful system on its own.

SECTION 3

EXAMPLES OF FIELDIED EXPERT SYSTEMS

To give the reader a better sense of how expert systems are currently being employed, this section presents brief descriptions of some fieldied expert systems. The selected examples are listed in table 3-1. Although C³I examples are more relevant to the needs of ESD readers, there are currently no truly operational expert systems for C³I applications--in fact, there is a paucity of fieldied systems in any field. Therefore, examples have been drawn from a variety of fields, from wherever working expert systems can be found. The systems listed in the table are described below. A general reference follows each description.

3.1 ACE

The Automated Cable Expertise (ACE) system is designed for troubleshooting and maintenance of telephone cables. ACE is an automated analysis system that digests hundreds of telephone maintenance reports daily. These reports are provided by the Cable Repair Administration System (CRAS), a conventional data base management system. ACE is activated nightly to study the day's reports and then send messages on troublespots to the ACE user via electronic mail.

The main sources of knowledge used to construct ACE were textbooks on telephone cable analysis, advice from the developers of CRAS, advice from theoreticians in Bell Telephone Laboratories, and users of CRAS performing the actual analyses. The knowledge base consists of all the relevant domain-specific knowledge required to diagnose cable faults. The inference engine controls the deductive process and is represented in rule form. Both are written in Lisp and the OPS4 Production System. The CRAS system provides the data base. ACE, then, represents a merger of expert system technology with older data base management technology.

ACE has been field tested since the spring of 1982. Analysts are reported to be satisfied with ACE's performance (Vesonder, et al., 1983).

Table 3-1. Examples of Fielded Expert Systems

<u>Name of System</u>	<u>Developer</u>	<u>Problem Addressed</u>
ACE	AT&T	Troubleshooting and making recommendations for telephone cable maintenance
DENDRAL	Stanford U.	Identifying chemical compounds
MACSYMA	M.I.T.	Performing differential and integral calculus
MYCIN	Stanford U.	Diagnosing and treating infectious blood diseases ¹
ONCOCIN	Stanford U.	Monitoring out-patient oncology treatment
PROSPECTOR	SRI	Finding mineral deposits
PUFF	Stanford U.	Assessing respiratory function
SOPHIE	Bolt Beranek and Newman, Inc.	Tutoring electronics troubleshooting
XCON	Digital Equipment Corporation	Configuring computer installations

¹At present, MYCIN is not used on wards, primarily because of its incomplete knowledge of the full spectrum of infectious diseases (Barr, et al., 1982, vol. 2, p. 192). The system is included because it is fully developed; it has compared well with physicians in formal evaluations, and because its domain-independent core (that is, EMYCIN or Essential MYCIN) has been used to build other systems. An example of a non-medical application that was developed using EMYCIN is SACON. SACON advises structural engineers on the use of a very complex computer program, MARC, which simulates the response of mechanical structures to various load conditions.

3.2 DENDRAL

The DENDRAL system infers the plausible structures of unknown organic compounds by using formulae and mass spectrograms. DENDRAL enumerates every possible organic structure that satisfies the constraints apparent in the data and rapidly eliminates implausible structures. Because DENDRAL systematically generates all plausible structures, it finds even those structures that humans overlook. DENDRAL surpasses all humans at its task (Hayes-Roth, et al., 1983, p. 9) and is consulted by chemists from all over the world. DENDRAL, about sixteen years old, is a Stanford project (Lindsay, et al., 1980).

3.3 MACSYMA

MACSYMA is a large, interactive computer system designed to solve a variety of mathematical problems. It performs algebraic simplification and it does differential and integral calculus symbolically. MACSYMA is very efficient and produces high quality results, and its performance excels that of most human experts.

MACSYMA is used extensively by engineers, mathematicians, and physicists. Many of the users spend a substantial portion of every day logged into the system. MACSYMA runs on a Digital Equipment Corporation (DEC) computer system at MIT, and is accessible via the ARPANET. A version for DEC's VAX computer is known as VAXIMA; a version programmed in Lisp is known as Edward (Mathlabs Group, 1977).

3.4 MYCIN

MYCIN is one of the oldest and best known expert systems. It is designed to provide computer-based medical consultations for infectious diseases. Specifically, MYCIN can diagnose and recommend therapy for blood infections and meningitis. The system was developed at Stanford University as part of the Heuristic Programming Project and was completed in 1969.

MYCIN interviews the physician-user for information about the patient. For example, it may ask whether the physician has obtained positive cultures from the site at which the patient has an infection. It couples this information with information from its knowledge base to infer a diagnosis and recommend a therapy. Medical knowledge is encoded as production rules. Conclusions are

asserted with an associated confidence factor. MYCIN is able to explain its reasoning and the user can add or modify rules. MYCIN uses approximately 450 rules.

MYCIN is equal in performance to nationally recognized experts in diagnosing blood infections and meningitis. However, it is not actually used on the wards since it lacks knowledge about related diseases. Thus, it has limited value for routine clinical use (Shortliffe, 1976).

3.5 ONCOCIN

ONCOCIN is an oncology decision advice system that was developed at Stanford University. The system monitors the treatment of oncology out-patients on experimental treatment regimens. ONCOCIN and PUFF are the only two medical expert systems that are used routinely by physicians (Clancey and Shortliffe, 1984, p. 16). The system is actually a set of programs, one of which is a rule-based reasoner that encompasses the necessary knowledge of cancer chemotherapy. The system also includes an on-line data base of patient information (Shortliffe, et al., 1981).

3.6 PROSPECTOR

PROSPECTOR is a computer-based consultation system developed at SRI International to assist a professional geologist in the early stages of investigating the exploration site or "prospect." The system makes probabilistic interpretations of soil and geologic data.

The program contains a knowledge-acquisition system (KAS) that facilitates the acquisition of knowledge by prompting the user. PROSPECTOR matches data from a particular situation against models that describe a large number of disjoint classes of situations. These models are formal descriptions of the most important types of ore deposits. The data are geological observations. Since these data are assumed to be incomplete, conclusions are expressed as probabilities or degrees of match. Inference rules specify how the probability of one assertion affects the probability of another assertion.

The system performs well. PROSPECTOR has predicted several deposits whose existence was subsequently confirmed by drilling, including a molybdenum deposit worth \$100M (Duda, et al., 1978).

3.7 PUFF

The basic task of PUFF is to interpret the results of pulmonary function tests. It produces a diagnosis and a set of interpretations and conclusions similar to those a physician would produce if given the same initial data.

PUFF is in actual use at the Pacific Medical Center Hospital in San Francisco. The system serves as a practical assistant to the pulmonary physiologist. Each PUFF report is read by a physician. More than half of the reports are accepted without any change and most of the remaining reports require only a few additional comments (Miller, 1984).

PUFF was developed from research done on the MYCIN system, and it was built using a generalization of MYCIN known as EMYCIN or Essential MYCIN (van Melle, 1979). (EMYCIN is discussed as a system building tool in section 3.) PUFF is made up of the EMYCIN programs and a knowledge base about pulmonary disease. PUFF employs a set of about 55 rules (Kunz, et al., 1978).

3.8 SOPHIE

SOPHIE (a SOPHisticated Instructional Environment) is a system for intelligent computer assisted instruction (ICAI) in the context of a simulated electronics laboratory. The system was developed at Bolt Beranek and Newman, Inc., over more than a decade ago.

The problem for the student is to diagnose the faults in a malfunctioning piece of equipment. The student acquires problem-solving skills by trying out ideas rather than by instruction. SOPHIE randomly selects a fault and inserts it into a simulation model of the circuit. The student is given a diagram of the circuit and begins troubleshooting by performing measurements.

In addition to domain-specific knowledge for problem-solving, SOPHIE contains numerous domain-independent inferencing mechanisms to answer the student's questions, criticize student's hypotheses, and suggest alternative hypotheses. The system also judges the student's suggestions for new measurements.

Extensions to SOPHIE produced SOPHIE-II. This system includes a troubleshooting game with two teams of students, and an articulate explainer. The explainer can articulate its deductions and provide global strategies to guide troubleshooting (Brown, et al., 1974).

3.9 XCON

One of the best known expert systems in use is a four-year-old program developed by groups at Carnegie-Mellon University (CMU) and the Digital Equipment Corporation (DEC) to configure its VAX computer. Most VAXs are custom tailored. This tailoring poses time-consuming, complex configuration problems.

XCON plans arrangements of components for VAX super-minicomputer systems and PDP. For example, XCON checks for correct cable lengths and required memory size. It also prints a diagram to assist with assembly.

XCON was built using a software tool called OPS5 (see section 4). XCON's knowledge base grew over time as rules were added, deleted, or modified as needed. From an original 200 rules, the system now contains between 3,000 and 4,000 rules.

According to trade literature, XCON configures the most complex order in less than one minute. The system is less error-prone than most DEC technicians (Abramson, 1984).

SECTION 4

TOOLS FOR BUILDING EXPERT SYSTEMS

Until recently, almost all expert systems were built from scratch, usually as part of a research effort. This required the builder to be highly trained in the concepts and techniques of artificial intelligence. Such people are scarce and, when found outside academia, usually very expensive. The shortage of experienced personnel for building expert systems has limited the number of expert systems that can be developed.

Over the past few years, tools for building expert systems have begun to make their appearance, first from the research community and later from the commercial world. The use of tools has simplified the development of expert systems in some cases. They have proven especially useful in the building of quick, early prototypes of expert systems. However, these tools have also attracted overblown expectations as to what can be accomplished with them. Some vendors even claim that their tools make it possible for people with no background in AI to develop expert systems. It is no more realistic to believe that a good tool can replace experience in expert system development than in any other specialized software area. For example, a novice should not be made responsible for building a corporation's financial data base no matter how good a data base management system is available.

However, expert system building tools can be valuable for the right applications under the right circumstances. Choosing an appropriate tool for building a particular system is both a difficult and crucial task. This section is meant to highlight some of the considerations in choosing a tool, but is not intended to provide definitive guidelines. This section will examine the tools available for expert system construction, and their features. A subset of these tools is described briefly.

Before beginning the discussion of specific tools, the role of tools will be discussed in section 4.1. A brief overview of the languages of artificial intelligence will be presented in section 4.2. Nearly all the tools have been written in one of these languages, and use of the tool often requires knowledge of the underlying language. Specific tools are described in section 4.3.

4.1 THE ROLE OF TOOLS IN BUILDING EXPERT SYSTEMS

There are three approaches one can take in building an expert system. The first approach, that of the past ten years, looks at the expert system program as a research project that requires individuals well schooled in the concepts of artificial intelligence. Nearly all expert systems to date have been built this way and nearly every such project has uncovered new AI techniques to make the research approach worthwhile.

The second approach begins with an existing expert system. An attempt is then made to separate the application specific part of the system from the rest of the system. This approach may be very useful if the system to be built is similar in nature to an existing system. For example, the nature of the problem of medical diagnosis is not very different from the nature of the problem of electronic system diagnosis. The purpose of a diagnostic system is to infer the cause of a problem given its symptoms. On the other hand, two systems for weapons analysis will not be similar if one does logistics planning and one does effectiveness analysis. Although these two systems have the same domain, that is, weapons analysis, they address different problems.

The second approach suggests still another approach; namely, to begin with a set of tools designed to aid in building expert systems. Recently, both universities and private corporations have generalized their research into tools for the development of expert systems. These tools are to expert systems much as Adabas and its contemporaries are to the field of data bases. They offer the basic constructs necessary for building a system, but they still require developers to choose the appropriate construct for their applications, and to gather, organize, and enter all the data and knowledge upon which the program will act.

4.2 FEATURES OF EXPERT SYSTEM BUILDING TOOLS

Much as there are different data base organizations--relational, hierarchical, network--whose use depends on the relationships among the data and particular goals of the given application, so, too, in expert systems are there a number of different technical approaches available for reasoning in the inference engine. The popular approaches so far developed include forward chaining and backward chaining. How these are implemented is not important at this point; what is important is to realize which approach is more appropriate to a particular type of problem. Forward chaining starts with the evidence or current facts and reasons forward until it reaches a final goal state. Forward chaining is useful in problems where there

are many possible acceptable but rather different solutions to a problem. For example, in design problems where the nature of the problem is to configure objects under constraints, there may be many acceptable design solutions that satisfy the constraints. An example of a system that is a classic forward-chainer is R1, later known as XCON, a system for configuring VAX computer systems from customer requests.

Backward chaining is another inference mechanism that starts from a possible conclusion and works backwards to see if it can justify that conclusion based on the starting evidence. Backward chaining is useful in problems where there is only one acceptable solution (or relatively few solutions) to a problem. For example, in diagnosis problems the purpose of the analysis is to find the cause of the problem so that a cure may be effected. An example of a system that is a classic backward-chainer is MYCIN, a system designed to diagnose and treat infectious blood diseases. These approaches are by no means exclusive or exhaustive. Most tools currently available use them in conjunction with one another and with other methods as well.

In addition to the inference engine, an expert system contains a method for knowledge representation. The most common method is to use rules, that is, a series of if-then clauses that are executed, or fired, when they are ready (see section 2). Rules contain domain knowledge and the "rules of thumb" that experts tend to use as aids to problem solving.

A strategy for which rule will be chosen from among those ready to fire is an important part of a system. As explained in section 2, such a strategy performs what is known as conflict resolution. Some strategies favor rules with the most complex antecedents; other strategies favor rules with the most recently used antecedent; other strategies simply use the rule that appears first in the data base.

Some systems allow the builder to add a certainty factor to each rule. A certainty factor is a measure of the expert's confidence in the validity of the rule. Some tools allow the builder to choose from among a number of different control strategies or to implement one of the builder's own design. Full control flexibility may be useful in some applications, but unnecessary in others.

In conjunction with rules, some tools allow knowledge to be set up in a network of relationships (see section 2). Knowledge components are allowed to inherit features from related components so that information does not have to be repeated for every component

in the system. In such a system, to say that a Camaro is a kind of car would mean, roughly speaking, that things the system "believed" about cars it would also "believe" about Camaros. Such systems use semantic nets or frames as their knowledge representation method. However, even within the frames, rules are normally applied to develop the inferences necessary to solve the problem at hand.

Expert system building tools are constructed with one or more of these inference mechanisms and knowledge representations, just as data base building tools are built around a particular data organization. In addition, other useful facilities may be supported by the tool.

Some tools provide an explanation facility. Such a facility may range from merely explaining why the system chose a particular rule to fire from among a group of rules that were ready to fire, to explaining how the system came up with a particular solution to a problem.

Some tools support what is known as active values. These have the following role. The builder can specify that when a particular value in the system changes, a builder-defined function should be run or activated. For example, each time information is entered or is inferred by the system causing a variable named "temperature" to change, a function might be run to update a thermometer display.

Some expert systems incorporate what might be thought of as a kind of "learning." In such a system, the "beliefs" the system has inferred while trying to solve a particular problem are added to its knowledge base permanently, not just for the run of the problem. In this way the system gets "smarter" each time it attempts to solve a problem. The major difficulty with such a scheme is that "facts" may change over time. For example, the fact "Iran is an ally" was true but is no longer true. Some expert system tools support what is known as truth maintenance or belief revision or non-monotonic reasoning. An expert system that might find such a facility useful is one that is expected to change over time as new information, or perhaps more refined information, is added to it. "Beliefs" that were developed based on older information must be brought into question when new information that contradicts or qualifies the old information is introduced into the system. In such a system the chain of reasoning that led to a "belief" must be remembered, so that "beliefs" that were based on the old information can be tracked down and removed from the knowledge base.

Certainly an important aspect of any computer system is the user environment. Many of the recent tools have elaborate user interfaces. They may use a "mouse," they may use a natural language

for entering information, and they may employ icons, windows, graphics, and a natural language for displaying information.

Another consideration in choosing a tool should be how well the tool is integrated into its environment; that is, how easily can a user augment or alter its behavior. At a minimum, for example, can a builder-defined Lisp or Fortran function be called by the system?

It is also important to consider the amount of support for the use and maintenance of the tool the purchaser can expect to receive. In dealing with expert systems, questions of support are vital due to the immaturity of the product. Research tools are not known for their support to naive users. Many tools were developed as doctoral theses and have since been augmented as necessary. When a copy of such a system is acquired, it is the user's responsibility to keep it current and write all the necessary support software to make it useful. The advantage of these systems, however, is their low cost. They are either free or available for a small licensing fee. On the other hand, some of the commercial products offer classes and consultation included in the price of the system, but are expensive.

Another important consideration in choosing a tool for building an expert system is who is going to be the user or what is the system builder's background. Is the builder a researcher in artificial intelligence technology? Does the builder have previous experience in building an expert system? Or is the builder a domain expert who is serious about learning how to use expert systems technology? Many of the tools advertise that they can be used by anyone with no previous experience in anything. Obviously, the buyer should beware.

Finally the reader should be strongly reminded that no one of the tools listed here or elsewhere will meet everyone's needs at all times. Nor is that likely to change in the near future (consider the number of different programming languages in existence). Some of the features of the tools discussed below may be useful in solving certain problems and absolutely useless in solving others. The problem must be analyzed carefully to determine which features would be critical, which would be useful, and which would be useless. In what is to follow, we will present a number of different tools that are currently available. It is not an exhaustive set. However, we believe it is a reasonably representative set.

4.3 PROGRAMMING LANGUAGES FOR WRITING EXPERT SYSTEMS

We begin our discussion of expert system tools with a discussion of three of the most popular artificial intelligence languages: Lisp

(list-oriented), Prolog (logic programming) and Smalltalk (object-oriented). The commonality of these languages is their capability for symbolic manipulation.

4.3.1 Lisp

Lists, trees, and networks are useful data structures for representing complex relationships between objects. Lisp deals with such data structures as primitive entities; they come naturally to Lisp. Another important strength of Lisp is that programs may be treated as data and data may be treated as programs. Given a certain input, a Lisp program may use that input to create a function and then proceed to execute that function in the same Lisp world. This capability is usually considered a flaw from a software engineering point of view. But it allows the programmer to handle what is expected to happen in the run of a program in a very flexible way. The interactive program development environment has also contributed to its popularity. Traditional computer hardware was designed to support "number crunching" applications. Lisp did not perform particularly well on such machines and, therefore, acquired a reputation for being slow and expensive to run because it used a lot of system resources. Recently, computer systems have appeared on the market whose logical architecture is specifically designed to support the economical development of artificial intelligence programs. Many of the expert systems built to date have been written in Lisp, even though Lisp has no built-in inference engine.

4.3.2 Prolog

Prolog describes known facts and relationships about a problem. The Prolog program consists of a set of clauses, in which each clause is either a fact about the given information or a rule about how the solution may relate to or be inferred from the given facts.

There are a number of reasons for the growing interest in Prolog. The inference engine for backchaining is a part of the language. Prolog can be used as a relational data base query language because it supports a subset of predicate calculus. As in Lisp, it is possible to treat data as program, and program as data. An interactive program development environment is provided. Prolog also seems to hold promise to support massively parallel computation. This is one of the aspects of Prolog that led the Japanese to become interested in exploring its use for 5th generation architectures.

4.3.3 Smalltalk

In Smalltalk, the programmer divides the world into objects. Each object maintains its own data representation and processing code hidden from all other objects, and is thus independent of them. Objects may request information or processing from other objects through messages. Objects can inherit capabilities from other objects within the same class, so that functions do not have to be copied for every type of object. For example, an object representing an airbase could receive messages requesting allocation of aircraft. The methods by which it performed this function would be hidden from all other objects, the object would simply return the final answer. Some of its methods, such as determining the distance to a destination, may, in turn, be inherited from another object class, such as that of stationary physical entities. Object-oriented programming supports modular programming and reusable code.

4.4 DESCRIPTION OF TOOLS

Several expert system building tools are described in this section. The description of each tool begins with an overview paragraph and follows with paragraphs on technical content, user facilities, and aids for the knowledge engineer.

The expert system building tools that are discussed, plus some additional tools, are listed in table 4-1 with their manufacturers. Some of their key features are summarized in table 4-2. This list is not exhaustive, but does contain many currently available tools. It is intended to give examples of the more popular systems that are available ranging from free unsupported university tools to \$60,000 commercially supported tools. Two new tools, ART and SRL, were still in beta testing as of the time information was collected and will not be discussed here. Most of the systems described below are written in Lisp.

4.4.1 EMYCIN

EMYCIN or Essential MYCIN was developed at Stanford in the late 1970s, and along with OPS5 (see below) is a grandparent of expert system building tools. It was generated from an expert system, MYCIN, that addresses the problem of diagnosing and treating infectious blood diseases. (See section 3 for a description of MYCIN.) EMYCIN contains all of MYCIN except its knowledge of infectious blood disease and can facilitate the development of diagnostic applications.

Table 4-1. Expert System Building Tools: General Information

	Developer	Price (\$)	Computers
EMYCIN	Stanford Univ.	< 500	VAX VMS/UNIX
MRS	Stanford Univ.	< 500	Symbolics, VAX
OPS5	Carnegie-Mellon	< 500	VAX VMS, Symbolics,
Xerox			
YAPS	Univ. of Maryland	< 500	VAX UNIX
ROSIE	Rand Corp.	< 500	VAX UNIX, Xerox
LOOPS	Xerox	< 500	Xerox
AGE	Stanford Univ.	< 500	PDP-10/20, Xerox
KES	Software Arch. & Eng.	25K	VAX, Apollo, etc.
TIMM	General Research	40K	IBM, DEC, etc.
TIMM-PC	General Research	10K	IBM PC-XT
DUCK	Smart Systems Technology	6K	VAX, Apollo, Xerox, Symbolics
KEE	IntelliCorp	60K	Xerox, Symbolics
S.1	Teknowledge	50K	Xerox
M.1	Teknowledge	13K	IBM PC
Expert- Ease	Intelligent Terminals, Ltd.	2K	IBM PC


Source: The information in this table was gathered from trade literature and conversations with manufacturers and developers. We have attempted to be as accurate as possible.

Note: Cost figures were current as of September 1984. Most companies offer substantial volume discounts and some prices include training.


Table 4-2. Expert System Building Tools Characteristics

	EMYCIN	MRS	OPS 5	ROSIE	LOOPS	AGE	KES	TIMM	DUCK	KEE	S. 1	M. 1	Expert-Ease
Back-chain													
Forward-chain													
Inheritance													
Active value								?			?		
Certainty factors													
Belief revision													
Explanation													
Flexible control									?				
Integrated environment								?					
Course													
Guidance													
AI background required ?													

Key:  - Yes

 - No

 - Some or partial

 - Insufficient information
supplied by the developer

EMYCIN is a classic backward-chaining rule based system. All control is done through the firing of rules in a goal directed manner. As such, it initially selects one goal conclusion, analyzes all rules that verify that conclusion, then tries to prove the antecedents to that rule. Proving the antecedents usually involves establishing them as subgoals and trying to find rules that have them as conclusions. In EMYCIN this process goes on until the original goal conclusion has been shown to be substantiated by the facts or there are no rules that can fire. This exhaustive search is a conservative approach, both because the rules and facts are probabilistic in nature and because in life-threatening situations it is best to have as much evidence as possible to back up a conclusion. Knowledge is stored as rules.

EMYCIN is to be used in a Lisp environment and can make calls to Lisp. EMYCIN has an explanation facility that gives a trace of why a rule was fired or why a question was asked. This trace shows the procession of rules fired from the initial goal. A rule editor is also provided for checking consistency among rules and for checking for valid syntax. In addition, there are system-specific tracing and debugging tools. As is customary with university developed products, EMYCIN has no support, but it does have a detailed user's manual.

4.4.2 MRS

MRS was developed at Stanford in the early 1980s. It is unsupported and available for a nominal fee. MRS provides a toolbox for an experienced knowledge engineer to build a custom expert system. It is not a complete system, but is intended to be inserted in Lisp code as function calls. In its tool kit are mechanisms for backward and forward chaining with optional saving of intermediate conclusions and truth maintenance to handle non-monotonic reasoning (reasoning under retraction of earlier beliefs). MRS provides the builder with a lot of control over the strategy used for conflict resolution and searching. For example, the builder can choose from depth-first, breadth-first, or best-first search strategies, or can design other strategies. MRS supports these capabilities by using meta-knowledge, or knowledge about how MRS itself works. The knowledge is entered in the form of rules.

Since MRS was developed as a research project concerned with knowledge representations, little concern was given to a friendly system builder environment. Knowledge entry and editing is accomplished through the standard Lisp facilities of the particular machine being used. MRS has a justification facility to show the rules that justify a conclusion the system has reached.

4.4.3 OPS5

OPS5 is the most widely used expert system building tool. It was the basis for XCON, the DEC VAX configurer that is one of the best known commercial expert systems. OPS5's simplicity has allowed it to be ported to nearly all computers. Although it has no elaborate input/output or explanation facilities, it remains popular because of its straightforward inference mechanism and relatively large user base. OPS5 is available both in unsupported university versions and in well supported systems from commercial firms.

Just as EMYCIN is the classic back-chaining system, OPS5 is the classic forward-chaining system. Its inference strategy is quite simple. It contains a data base that contains the known state of the world as objects with associated attribute-value pairs. All rules are searched to find those whose antecedents are all known to be true. If more than one rule is found to be applicable, a conflict resolution strategy is then invoked to select a single rule. The chosen rule is fired, its conclusions instantiated into the data base, and the process repeated until some goal is reached or no rule can fire. Builder defined functions may be called from within OPS5.

4.4.4 YAPS

YAPS, which stands for Yet Another Production System, is an adaptation of OPS5 from the University of Maryland. It has the same structure as OPS5 but overcomes some of its shortcomings by allowing more complex structures in the data base.

4.4.5 ROSIE

ROSIE, the acronym for Rule Oriented System for Implementing Expertise, represents Rand's entry into the world of expert system building tools. It, too, is available very inexpensively and is unsupported. Rules may be entered in an English-like natural language as opposed to a Lisp-like structure (unlike OPS5, whose syntax is much like Lisp).

ROSIE's inference engine uses pattern matching to access its rule. Its commands are coded in structured English-like sentences, giving it the appearance of a relational data base that analyzes and acts on the rule base. The English statements also give a procedure-oriented style to rule access. There is a concept of inheritance in ROSIE. It also has the ability to communicate with

other systems or programming languages. It has no explicit explanation facilities. ROSIE uses the Interlisp editor for knowledge base entry.

4.4.6 LOOPS

LOOPS was developed by researchers at Xerox Corporation's Palo Alto Research Center. It is a complete package designed to run on the Xerox Lisp Machine. Although LOOPS is currently not a Xerox product, it is made available to Xerox Lisp machine owners for a nominal fee, and training classes are available for it. LOOPS currently supports only a forward-chaining inference engine. Object-oriented programming is supported, as well as data-oriented programming, via an active value mechanism.

The LOOPS environment makes use of the facilities provided by its host, a Lisp machine. The user enters data via the mouse and its related menus and windows. Icons and bit-mapped graphics, along with text, are used to display information. Questions can be asked at any time, either freely or through menus.

Debugging is provided with the Lisp machine's facilities. During any portion of the run, the system can be halted and investigated with the debugger, which uses textual and graphic displays. Additionally, windows can be created, allowing the programmer access to source code or to other files while the display remains on the screen.

4.4.7 AGE

AGE, short for Attempt to Generalize, is a collection of tools for building expert systems. AGE is an experimental system being developed by the Heuristic Programming Project at Stanford University. AGE is a research tool that is still under development. The AGE-1 system is written in InterLisp and currently runs on the DEC PDP-10 and DEC system-20 computer systems. A separate AGE-1.5 system is available for Xerox 1100 series workstations. AGE is available for a one time, royalty free distribution fee of \$500.00.

AGE has isolated several inference, control, and representation techniques from previous expert systems and has reprogrammed these modules for domain independence. AGE is a tool for building expert systems but is itself an expert system since it guides users to use

the modules to construct their own expert systems. It supports both backward-chaining and forward-chaining inferencing, as well as reasoning with uncertainty.

4.4.8 KES

There are three distinct inference engines in KES (Knowledge Engineering System) from which the designer can choose. One is a backward chainer, one applies Bayes' theorem, and one performs hypothesis and test. They can all handle incomplete and uncertain knowledge. Knowledge is stored in an attribute hierarchy schema.

The knowledge engineer uses a standard editor to enter the rules, attributes, and actions. The rules are compiled before the KES system can be run. The rules are examined for problems such as conflicting rules and unused decision attributes. The actions are the procedural portion of KES telling the system when to collect information and what goals to try to solve. The system then generates any subgoals or other actions necessary to solve the given problem. KES is designed to run on a VAX. Another version (called MKES) is available for an IBM PC.

4.4.9 TIMM

TIMM is primarily a decision support tool. Its knowledge is entered through situation formulations from which it deduces rules of inference. A domain expert provides a situation formulation by specifying values for attributes and the decision that is to be returned by the expert system under these circumstances.

If TIMM is not provided with enough situation formulations to detect a pattern of differentiation, it will present a set of attribute values to the user and ask for a decision selection under these circumstances. TIMM checks the knowledge base for consistency and completeness to ensure there are no conflicts. The rules are maintained in a frame-like system, allowing for inheritance and the use of daemons (functions called to produce desired side effects).

The end user of the expert system is asked multiple choice questions. Depending on the user's choice, either terse or verbose English statements will be presented. TIMM has no direct explanation facility. TIMM can be set up to link to knowledge bases in other TIMM systems so that a decision process can be broken down into smaller pieces.

4.4.10 DUCK

DUCK provides a logic programming environment within a Lisp environment. It is written in NISP, a generic Lisp, and is thus portable to many machines.

DUCK's inference engine uses both forward chaining and backward chaining within its logic programming approach. The combination of forward and backward chaining takes place concurrently, depending on the current state of the inferences. Such a combination is called opportunistic scheduling, a technique from recent research. The designer also can give direction as to how the search process should take place. In addition, the builder can write Lisp functions to be called by DUCK at any time.

Interaction with DUCK is via standard Lisp processes. DUCK's explanation facility allows the user to request partial deductions and watch the execution of the steps through the knowledge base. Debugging and knowledge entry are done through whatever LISP debugging capabilities are available on the specific machine.

4.4.11 KEE

KEE, the acronym for Knowledge Engineering Environment, promises very strong customer support. Of its \$60,000 purchase price, \$30,000 is for training, phone and on-site consultation. KEE runs on both the Xerox and the Symbolics Lisp machines. It has a sophisticated, quite friendly expert system building environment that makes extensive use of the Lisp machine's windows, menus, bit-mapped graphics, and "mouse."

KEE's inference engine contains both a backward chainer and a forward chainer. It gives the designer the ability to call builder defined Lisp functions through its active value mechanism. It does not have a built-in mechanism to handle uncertainties about knowledge. Knowledge and rules are stored in frames that have multiple lines of inheritance. This feature gives KEE an object-oriented flavor. The frames are executed according to both the needs of the inference engine and scripts and menus written by a designer.

KEE provides a debugger. During any portion of the run, the system can be halted and investigated with the debugger which uses textual and graphic displays. Additionally, windows can be created, allowing the programmer access to source code or other files while the display remains on the screen. Knowledge is entered through the Lisp editor.

4.4.12 S.1

S.1, a product of Teknowledge, has been available for about a year. It is highly integrated into the architecture of the Xerox Lisp Machine. It makes use of all of the features for windows and graphics that this machine provides. Courses and telephone support are available.

The inference engine of S.1 is a backchainer that handles uncertainties. Knowledge may be represented through both frames and rules, and certainty factors may be associated with rules. Builder defined functions may be called from within the S.1 environment. It has a full explanation facility that will respond to questions of why it has asked questions of the user and how it has reached its conclusions. Knowledge is entered through the Lisp editor.

4.4.13 M.1

M.1 is a member of a family of knowledge engineering products developed by Teknowledge. (See also S.1, above.) M.1 was designed as a software tool for exploring practical applications of knowledge engineering on an IBM PC. The system was designed for developing small scale operational systems where small is defined as an application that requires up to 200 knowledge base entries.

M.1 has a backchaining inference engine and a capability for establishing a set of high priority goals. Certainty factors can be associated with rules. There is a capability for segmenting knowledge bases.

The price of M.1 is \$12,500. The price includes the tuition for one participant in a four-day hands-on training course. An extensive reference manual is supplied. In addition, several annotated sample knowledge systems are provided on a disk. These systems can be demonstrated or modified.

In building a system with M.1, an English-like language is used to state facts and rules about the chosen application. A knowledge base is then created using any standard text editor. After the knowledge base is developed, M.1 can engage the user in a question-and-answer dialogue and use the knowledge base to provide a recommendation or conclusion. The user can request explanations during or after the reasoning process.

4.4.14 EXPERT-EASE

Expert-Ease is primarily a decision support system. It synthesizes into rules situation formulations that are provided by the builder. A domain expert provides a situation formulation by specifying values for attributes and the decision that is to be returned by the expert system under these circumstances. Expert-Ease makes use of a spreadsheet format for collecting knowledge from the builder. It collects data from end users by guiding them through a series of multiple-choice questions.

The system runs on an IBM PC and on certain IBM compatible machines. No support is available, but the builder/user's manual is extensive and well written. However, only small applications of very limited types are suitable for this tool.

SECTION 5

DARPA'S STRATEGIC COMPUTING PLAN

In October 1983, the Defense Advanced Research Projects Agency (DARPA) released its Strategic Computing Plan¹ (SCP). The Plan sets forth an important new program to use recent advances in artificial intelligence, computer science, and microelectronics to create from FY84 through FY93, a new generation intelligent computing technology that will have unprecedented capabilities.

This section states the goals (section 5.1) and gives an overview of the SCP (section 5.2). Much of the SCP is concerned with extending the capabilities of expert systems technology and developing increasingly sophisticated expert systems for military applications. Expert systems technology is discussed in section 5.3. The major applications of interest, namely, autonomous vehicles, pilot's associates, and battle management systems, are described in section 5.4. The requirements for advanced hardware are summarized briefly in section 5.5. The management, schedule, and cost of the program are summarized in section 5.6. Section 5.6 also describes the Air Force's plans to coordinate activities described in the Air Force Master Plan for Research and Development in Artificial Intelligence with the SCP.

5.1 GOALS

The overall goals of the SCP are "to provide the United States with a broad line of machine intelligence technology and to demonstrate applications of the technology to critical problems in defense" (SCP, p. ii). The plan states that "this technology promises to yield strong new defense systems for use against massed forces, and thus to raise the threshold and decrease the chances of major conflict" (SCP, p. 10).

¹DARPA, Strategic Computing: New-Generation Computing Technology-A Strategic Plan for its Development and Application to Critical Problems in Defense, 28 October 1983.

5.2 OVERVIEW OF THE SCP

The SCP is an ambitious program to create a new generation of intelligent machine technology by building on recent advances in artificial intelligence, computer science, and microelectronics. The Plan extends from FY84 through FY93. Program costs have been estimated at \$300M for FY84 through FY86 with out-year funding to be determined as the Plan progresses. To accomplish its Program, DARPA will fund and coordinate research in government agencies, universities, and industry. DARPA will manage the Plan with close coordination with the Under Secretary for Defense, Research, and Engineering (USDRE) and the military services.

The program focuses on military applications that are challenging enough to stimulate technology development. Applications "pull" the creation of the technology base. That is, applications generate requirements for intelligent functions. Intelligent functions drive the requirements for system architectures whose requirements, in turn, drive the requirements for microelectronics.

Specific applications addressed in the Plan are autonomous vehicles, pilot's associates, and battle management systems. (Each is described in section 5.4.) These systems would be able to perform intelligent functions such as understanding natural language expressions, information fusion, machine learning, and planning. These systems would interact with their users through vision and the generation of visual images, and speech recognition and production.

Each system would contain one or more expert systems that will require significant developments in expert systems technology. Since expert systems are the subject of this paper, the technology is discussed in some detail in section 5.3. Briefly, these systems would be characterized by very large knowledge bases containing from 10,000 to 30,000 rules and ultra-rapid processing speeds. By comparison, the largest operational expert system described in section 2.4 (XCON) has less than 4,000 rules with most systems having fewer than 500 rules. Currently, operational expert systems do not have requirements for real time processing. The applications in the SCP, however, will require processing speeds ranging from 1/3 times real time to 5 times real time speeds for simulations in dynamic settings. Finally, some applications will require multiple coordinating expert systems. There are no existing operational systems that are coordinated.

For the development of expert systems, DARPA envisions the need for contractors to have two new types of Lisp machines (see section 3). One will be 10 times faster than current machines and one will

be a low power, compact version for use in experiments about applications. The development will be by industrial manufacturers. DARPA expects to supply this equipment to contractors beginning in FY87.

Most of today's computers are still single-processor von Neumann machines. The underlying technology is not advancing at a fast enough rate to provide the massive increases in computing power that will be required. Accordingly, in order to achieve the desired performance, the Plan calls for exploiting VLSI architecture and massive parallelism. Practical experience with parallel machines is still very limited.

Weight, space, and power requirements will impose additional constraints on the systems to be developed. There is a great deal of interest in small, rugged packages that can be used, for example, in cockpits. The Plan will rely on effective exploitation of state-of-the-art microelectronics to meet the requirements for decreased weight, volume, and power. Silicon technology will continue to be the mainstay, because of its maturity, but gallium-arsenide based microelectronics will need to be further developed for some applications such as those that require survivable, space-based electronics.

The output of the Plan is expected to be a large array of intelligent technology that can be applied to diverse military applications. In addition, DARPA hopes to stimulate massive amounts of technology transfer into the commercial sector.

5.3 EXPERT SYSTEM TECHNOLOGY

The specific issues relating to expert system technology that are discussed in the SCP are summarized in this section. According to the SCP, expert system technology is most appropriate for command and control operations, situation assessment, and high-level planning. Since currently the most time-consuming portion of the process of building an expert system is usually knowledge engineering, the expert systems component of the SCP emphasizes knowledge acquisition and representation.

Research in representation will be directed toward building a capability for very large knowledge bases containing up to 30,000 rules. To put this size into perspective, most current rule-based expert systems operate with fewer than 500 rules.

Knowledge acquisition will focus on developing facilities for automated input of domain knowledge directly from experts, text, and

data. Inference techniques will need to be extended to handle the enormous knowledge bases and will require improved capabilities for handling uncertain knowledge and inaccurate and incomplete data.

Explanation and presentation systems will require advances in speech recognition and production to allow for verbal commands and verbal data input from the user and for task-oriented conversations with an expert system. The initial efforts will focus on speaker-dependent isolated word recognition in a noisy environment. The ultimate objective is to produce a natural language subsystem that is interactive, that can accommodate multiple users and that understands streams of textual information.

Most military expert systems will be required to perform in near real time or in simulations at faster than real time. This requirement for speed will severely tax current computational resources. (Currently fielded expert systems operate in domains in which real time response is not a requirement.) DARPA estimates that systems will be required to perform at a capacity of 12,000 rule inferences per real time second at rates up to 5 times real time, and that because of size and cost constraints, the required architecture will be achieved using VLSI devices and massive parallelism. The resulting technology is expected to significantly advance the capability of expert systems, in industrial as well as military applications.

The SCP contains a detailed description of technological developments in expert systems to be accomplished from FY84 through FY93. The major functional capabilities and milestones are shown in table 5-1. With reference to the capabilities shown in the table, there are existing expert systems with confidence levels associated with conclusions. Improvements in sensors will include vision systems that take in data from imaging sensors and will interpret these data in real time. Speech input is rare today, but one of the goals of the SCP is to develop expert systems that accept commands and data input as isolated spoken words. Ultimately such systems will understand streams of textual input and will be able to carry on conversations with the user. So far, however, only one or two systems being developed for battlefield simulations recognize isolated spoken words as commands.

Finally, we know of no cooperating expert systems. In the SCP, the Pilot's Associate exemplifies a set of expert systems that will be designed to cooperate with each other. For example, during an engagement, knowledge bases can be updated by individual pilots as tactical events change. This newly "learned" knowledge will be exchanged automatically among Pilot's Associates.

Table 5-1. Strategic Computing Expert Systems Technology
Major Functional Capabilities and Milestones

	<u>Capabilities</u>	<u>No. Rules</u>	<u>No. RIPS¹</u>	<u>Time</u>	<u>Complexity²</u>
FY86	Situation assessment with confidence levels of conclusions	2,000	1,000	1/3-1/2 real time	Modest
FY90	Dynamic adaptation of expert systems with sensors and speech input	10,000	4,000	real time	Intermediate
FY93	Multiple cooperating expert systems with planning capabilities	20,000	12,000	5 x real time	High

¹RIPS - Rule inferences per second

²Complexity relates to the context in which the expert system technology will be employed

Source: DARPA, Strategic Computing, 28 October 1983, Appendix II.1.4.

5.4 APPLICATIONS

The overall goals of the SCP will be addressed by focusing on three specific military applications. They are autonomous vehicles (section 5.4.1), pilot's associates (section 5.4.2), and battle management systems (section 5.4.3). These applications stress different aspects of machine intelligence. Each is thought to demand an "aggressive but feasible level of functional capability" (SCP, p. v) and, hence, can provide a "pull" on the new technology. The applications were selected for their relevance to critical defense problems.

5.4.1 Autonomous Vehicles

The SCP defines autonomous systems as robotic devices that are able to sense and interpret their environment, to plan and reason using sensed data, to initiate actions to be taken, and to communicate with human beings on other systems (SCP, p. 21). Examples are "smart" munitions and cruise missiles. Completely autonomous air, land, and undersea vehicles will require the kinds of developments addressed in the SCP.

An autonomous land vehicle system is used as an example of a class of autonomous vehicles. Such a system would require an expert system for navigation and a vision system. The expert navigation system must do such things as plan routes using terrain data, estimate the vehicle's position, and generate moment-to-moment steering and speed commands. These functions must be accomplished in real time or near-real time while the vehicle is moving at speeds of up to 60 km/hr. According to the SCP, such an expert system is expected to require on the order of 6,500 rules firing at the rate of 7,000 rule inferences per second (SCP, p. 22).

The vision system must take in data from imaging sensors and interpret these data in real time to produce a symbolic description of the vehicle's environment. It is anticipated that an aggregate computing requirement of 10-100 BIPs (billion equivalent von Neumann instructions per second) will be needed to accomplish its tasks. This estimate compares with capabilities of 30-40 MIPs (million instructions per second) in today's most powerful von Neumann type computers.

There are also strict requirements for the computing system's weight, power, and space. A one to four order of magnitude weight, space, and power reduction over current computing systems will be required.

5.4.2 Pilot's Associate

The Pilot's Associate is an intelligent system to assist a combat pilot in the air and on the ground. It does not replace the pilot. Rather, it complements the pilot by performing special functions and lower-level chores so that the pilot can concentrate on tactical and strategic objectives. The Pilot's Associate is being conceived as a set of expert systems and natural interface mechanisms to operate in real time. The project is expected to complement the USAF Cockpit Automation Technology effort.

The Pilot's Associate will be a personal associate to a specific pilot. It will be trained by the pilot to respond in preferred ways if, for example, an engine fails during combat. Other features will include general knowledge about the aircraft, instructions on advanced tactics from more experienced pilots, and features for updating the knowledge bases. A novel feature is that certain classes of newly learned knowledge will be exchanged automatically among Pilot's Associates.

Nine knowledge bases will be developed for the Pilot's Associate. They are the aircraft/pilot, tactics and strategy, enemy aircraft, communication, geography, navigation aids, the mission, enemy defense, and friendly forces. The knowledge bases will be significantly larger than any previously constructed. They will contain several thousand rules that will have to be processed perhaps 100 times faster than rates that are attainable with current technology.

5.4.3 Battle Management System

A Battle Management System (BMS) is being conceived as a large, sophisticated expert system to aid in the management of large-scale defensive engagements. It could be integrated into various other defense systems on land, water, or in the air.

The BMS would interact with the user at a high level through speech and natural language. It would display a detailed picture of the battle arena, including force disposition, weather forecast, and so forth. It would generate hypotheses about possible enemy intent, assess the likelihood of each possible action, prioritize these according to their relative likelihood, and explain its prioritization. Using its knowledge bases of own force and enemy capabilities, it would generate alternative courses of action, evaluate the likely outcomes of each course of action, and evaluate the relative attractiveness of each alternative according to criteria that had been supplied previously.

Although the above activities may sound as if the BMS would perform Bayesian decision analysis, this is not the case. Rather, it would generate alternatives and project outcomes using a rule-based simulation operating in faster-than-real time.

The BMS would act as a capable assistant to the commander by presenting the information described above and explaining its reasoning. The commander would select a course of action and then the BMS would develop and disseminate the operation plan. At the conclusion of the engagement, the BMS would modify its knowledge bases based on results. In this way, the BMS would update itself based on its own experience.

A number of expert systems and a natural language interface will have to be developed to implement the desired capabilities of the BMS. The SCP estimates a distributed expert system that will require approximately 20,000 rules and processing speeds of 10 BIPs (billion equivalent von Neumann instructions per second). The natural language system will require a processing speed of about one BIP.

5.5 HARDWARE

In order to conduct successful military demonstrations of these applications, it will be necessary to develop new functional capabilities for machine intelligence, such as improvements in the understanding of natural language, signal interpretation, and information fusion. Although these capabilities are provided by software, they depend strongly on the underlying hardware and system architecture for speed and efficiency. Accordingly, a component of the SCP is a plan for the development of advanced machine architectures. The SCP also depends on the exploitation of faster, denser, more radiation-resistant, lower-power devices provided by state-of-the-art microelectronics.

5.6 MANAGEMENT, SCHEDULE, AND COST

The management of the SCP will be carried out by DARPA. Within the DOD, DARPA will coordinate closely with USDRE and the military services. (See next page for a description of activities planned by the Air Force.) A panel of the Defense Science Board has also been convened to make recommendations on how best to use the new technology within DOD.

The SCP is designed to continue through FY93. A summary of major efforts over the 10-year period for the expert systems

technology component was presented in section 5.3. Additional planning time lines may be found in the appendices to the SCP.

A cost summary is shown in table 5-2. Total costs for the fiscal years 1984 through 1986 are expected to be approximately \$300 million. Out-year funding levels are to be determined by the progress of the program.

Table 5-2. Strategic Computing Plan Cost Summary in \$Millions

	<u>FY84</u>	<u>FY85</u>	<u>FY86</u>	<u>FY87</u> ¹	<u>FY88</u> ¹
Total Military Applications	6	15	27	TBD	TBD
Total Technology Base	26	50	83	TBD	TBD
Total Infrastructure	16	27	36	TBD	TBD
Total Program Support	2	3	4	TBD	TBD
Total	50	95	150	TBD	TBD

¹Out-year funding levels are to be determined (TBD) by the progress of the program.

Source: DARPA, Strategic Computing, 28 October 1983, figure 6-2, p. 66.

To ensure coordination between the SCP and related service programs, the Joint Directors of Laboratories (JDL) has formed a technology panel on strategic computing (TPSC) as a standing panel of the JDL. The TPSC is forming seven subpanels to correspond with our areas of application and the three technology thrusts identified in the SCP. These subpanels, together with recommended lead organizations and Air Force members, are shown in table 5-3.

TPSC members will work with DARPA's SCP managers to achieve overall coordination. Technical coordination will be accomplished by having individual subpanel members interface with SCP thrust managers.

Table 5-3. Technology Panel on Strategic Computing
Joint Directors of Laboratories Subpanel Membership

<u>Subpanel</u>	<u>Recommended Lead Organization</u>	<u>Air Force Member(s)</u>
Battle Management	Navy	RADC
Pilot's Associate	Air Force	AFWAL
Autonomous Vehicle	Army	AFWAL
Space Based System	Undecided	AFSTC
Architecture	Navy	RADC & AFWAL/AA
Microelectronics	Army	RADC & AFWAL/AA
Software AI	Air Force	RADC

Key AFSTC - Air Force Satellite Tracking Center
AFWAL - Air Force Wright Aeronautical Laboratories
AFWAL/AA - Air Force Wright Aeronautical Laboratories
Avionics Laboratory

Source: Air Force Artificial Intelligence Research and Development Master Plan, 1984, p. 20.

SECTION 6

CONCLUSIONS AND RECOMMENDATIONS

This report was written to provide some insight into what expert systems are and what the current state of the technology is capable of doing in the area of C³I applications. A second goal was to help potential users of expert system technology understand what kinds of applications are good candidates for expert system solutions and to help them evaluate where expert systems could benefit their programs. In this section, we will review the important points raised in the paper and summarize our recommendations to those considering an expert system application.

6.1 CONCLUSIONS

1. Expert systems are intelligent assistants to human decision makers. They cannot, and are not designed to replace human experts in all aspects of their jobs.
2. Expert systems are good for solving problems where algorithmic solutions don't work well.
 - o where data is fragmentary, incomplete, or fuzzy,
 - o where solutions require reasoning with uncertain evidence,
 - o where there is no single "right answer,"
 - o where the system will be modified frequently,
 - o where solutions change with time and context,
 - o where the system must be highly flexible in the kinds of questions it can answer,
 - o where an algorithmic solution is too expensive to run.
3. There are only a handful of operational expert systems, none performing C³I applications. However, a large number of C³I expert systems are under development, and several have shown success in initial testing.
4. Knowledge engineering (knowledge acquisition and knowledge-based development) is usually the biggest bottleneck in expert system applications.
5. Building expert systems is an evolutionary process. The process should begin with a small-scale prototype and go

through several iterations until a satisfactory version is achieved. Even after an expert system becomes operational, it often continues to be modified and enhanced. Consequently, expert systems do not lend themselves to a fixed specification design--rather, the specification should evolve with the system.

6. The use of tools for building expert systems may speed the development process, especially during the prototyping stage. However, the value of these tools in implementing full-scale operational systems is largely untested. Moreover, a powerful tool is not a substitute for having some expert system experience within the system development team.

6.2 RECOMMENDATIONS

For those considering the development of an expert system application, we offer the following recommendations:

1. Choose the application wisely. Start with a small, well-bounded problem, ideally in a problem domain where expert system technology has already been used successfully. Make sure at least one domain expert is available and willing to spend a good deal of time helping to develop the system.
2. Begin with a small-scale prototyping effort early in the development cycle (certainly within the first year) to determine the feasibility of the application and to lend insight into the requirements for the full-scale system. Be prepared to throw away the prototype and even some of the early incremental versions of the system. Expect the development to be highly evolutionary.
3. Allow a long lead time to develop the expert system. Past experience has shown it takes, on the average, four to five years to complete an operational expert system application. The use of expert system building tools may reduce development time to some extent, but don't count on it.
4. Consider the use of expert system building tools, especially as a way to get a prototype running quickly. The tool should be carefully matched to the requirements of the application, since different tools have widely varying features. But keep in mind that even with a good tool, the development team still needs some experience in expert systems to develop a good expert system.

LIST OF REFERENCES

1. B. Abramson, "Applied AI at Digital," The Artificial Intelligence Report, January 1984, pp. 3-6.
2. Air Force Artificial Intelligence Research and Development Master Plan, DCS Science and Technology, Headquarters Air Force Systems Command, May 1984.
3. A. Barr, P. Cohen, and E. Feigenbaum, eds., The Handbook of Artificial Intelligence, 3 volumes, Los Altos, CA: William Kaufman, Inc., 1982.
4. J. S. Brown, R. R. Burton, and A. G. Bell, "SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting," BBN Report No. 2790, Cambridge, MA: Bolt Beranek and Newman, 1974.
5. W. J. Clancy and E. H. Shortliffe, eds. Readings in Medical Artificial Intelligence: The First Decade, Reading, MA: Addison-Wesley Publishing Company, 1984.
6. DARPA, Strategic Computing, Arlington, VA, 28 October 1983.
7. R. O. Duda, et al., "Development of the PROSPECTOR Consultation System for Mineral Exploration," Final Report, SRI Projects 5821 and 6415, 1978.
8. R. O. Duda, and E. H. Shortliffe, "Expert Systems Research," Science, pp. 261-268, 1983.
9. F. Hayes-Roth, D. A. Waterman, D. B. Lenat, eds., Building Expert Systems, Reading, MA: Addison-Wesley Publishing Company, 1984.
10. J. Kunz, et al., "A Physiological Rule-Based System for Interpreting Pulmonary Function Test Results," Heuristic Programming Project Report No. HPP-78-19, Computer Science Department, Stanford University, 1978.
11. R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project, NY: McGraw-Hill, 1980.

12. W. A. Martin, and R. J. Fateman, The MACSYMA System, In Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, Los Angeles, CA, 1971, pp. 59-75.
13. Mathlabs Group, MACSYMA Reference Manual, Cambridge, MA, Massachusetts Institute of Technology, Computer Science Laboratory, 1977.
14. D. C. McCall, P. H. Morris, D. F. Kibler, and R. J. Bechtel, STAMMER2 Production System for Tactical Situation Assessment, Technical Document 2984, Vol. 1, "Design Description," Vol. 2, "Code," San Diego, CA, Naval Ocean Systems Center, October 1979.
15. R. K. Miller, ed., The 1984 Inventory of Expert Systems, Fort Lee, NJ: Technical Insights, Inc., 1984.
16. H. P. Nii, E. A. Feigenbaum, J. J. Anton, and A. J. Rockmore, "Signal-to-Symbol Transformation: HASP/SIAP Case Study," AI Magazine, Vol. 3, No. 2, (spring 1982), pp. 23-35.
17. E. H. Shortliffe, Computer-Based Medical Consultations: MYCIN, American Elsevier, NY, 1976.
18. E. H. Shortliffe, et al., "ONCOCIN: An Expert System for Oncology Protocol Management," Proceedings of the Seventh IJCAI, 1981, pp. 876-881.
19. J. R. Slagle, Symbolic Automatic Integrator (SAINT), Ph.D. Diss. Rpt. 5G-0001, Lincoln Laboratory, Massachusetts Institute of Technology, 1961.
20. A. vanMelle, Domain-Independent Production-Rule System for Consultation Programs, In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, pp. 923-925, Stanford, CA: Stanford University, Department of Computer Science, 1979.
21. G. T. Vesonder, S. J. Stolfo, J. E. Zielinski, F. D. Miller, D. H. Copp, "ACE: An Expert System for Telephone Cable Maintenance," Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 1983, pp. 116-121.
22. P. Winston, Artificial Intelligence, Reading, MA: Addison-Wesley Publishing Company, 1979, 2nd ed., 1984.

DTIC

END

4-86